Utdrag fra boken *Ekte Programmering* (2021) av Corneliusen & Julin.
Mer informasjon: www.ignorantus.com/ekte/

Nils Liaaen Corneliusen

Sjur Julin

Ekte Programmering

En bok om programmering, programmerere, programmer og populærkultur

Nils Liaaen Corneliusen og Sjur Julin har opphavsretten til teksten.

Deler av teksten er basert på engelske artikler skrevet av Corneliusen i perioden 2010-2020. Disse ble publisert på nettstedet www.ignorantus.com, som eies av Corneliusen.

Det henvises til kapittelet *Lisensinformasjon* for detaljer om opphavsrett til kode og bilder.

All kode ol. benyttet i boken finnes på følgende nettsted:

www.ignorantus.com/ekte

Alle henvendelser om boken kan rettes til

ekte@in-a-vision.com

Utgiver:

Ignorantus AS
Munkedamsveien 57
0270 OSLO

ISBN:

978-1-716-41204-2

"So, I'm taking my work underground. I can't let it fall into the wrong hands again."
- Dr. Lawrence Angelo i filmen *The Lawnmower Man* (1992)

Innhold

Introduksjon

Jeg leste en gang en artikkel kalt *The Importance of Revisiting Your Old Code*[1] av Kieran Jones som sa:

> *As a developer you are constantly learning and improving, or at least you should be (more on this in a future post). One of the hardest things to do is to look back at old projects with your now more experienced eyes and wonder what the heck you were thinking back then.*

Se på tre år gammel kode du har skrevet og fått betalt for. Er utsagnet ovenfor sant? Da kan du si opp jobben med en gang, og gå hjem og øve på programmering før du prøver det igjen. Du skriver ikke kode: Du lærer fortsatt grunnleggende programmering for arbeidsgivers regning.

Gode programmerere begynner ikke å programmere på universitetet. Det er en lang og vanskelig prosess. En god en har skrevet kode så lenge han kan huske. Han unngår programmeringstrender og eliminerer dem nådeløst når de kommer i veien. Han har en datamaskin hjemme som han bruker til å skrive kode for moro skyld, og kanskje en samling mindre maskiner som *Raspberry Pi*. Eller noen mikrokontrollere med noe tilpasset maskinvare, for eksempel en laser eller et kamera eller to. Her er en smertefull sannhet: Ikke alle blir gode programmerere etter å ha programmert i årevis, akkurat som ikke alle blir gode musikere etter å ha spilt gitar i årevis. Programmering handler om å forstå mønstre, og veldig få gjør det bra.

En dårlig programmerer, ofte kjent som en karriereprogrammerer, har følgende kjennetegn: De begynner å bruke datamaskiner sent og mestrer, i beste fall, et smalt utvalg av fagområdet. De liker møter og planlegging, men skriver sjelden, om noen gang, kode som er annerledes enn eksemplene i lærebøkene. De tror at arbeidet er over klokka fem og holder seg ikke oppdatert om programmering på fritiden. De bruker ikke egne datamaskiner til å programmere for moro skyld, bare for penger på dagtid. Og for å være politisk ukorrekt: Når slike programmerere er fraværende fra arbeidet over en periode på måneder eller år, så tror de gjerne at dataverdenen har stått stille i mellomtiden. Men sånn er det ikke, dessverre. Det er da de ofte finner sin endelige posisjon innen brukerstøtte, kvalitetssikring eller som ubrukelig mellomleder. Jeg har vært i denne bransjen i flere tiår og sett det skje igjen og igjen. Og det vil fortsette å skje.

"Best practices" og "programming patterns" (må ikke forveksles med faktiske mønstre) er konsepter som er skapt for å få sånne dårlige programmerere til å skrive gjennomsnittlig kode. De har til og med laget en metodikk for det. Og ikke bare en, det er flere. Å prøve å formalisere hvordan man skriver kode er like produktivt som å gjete katter. Med mindre alt du har er sauer, selvfølgelig. Før du vet ordet av det fylles kontoret opp med håndhevere av den *riktige og korrekte måten* å programmere på, og antall personer som gjør ekte programmering konvergerer mot null.

Saken er at disse metodene pleier å bli introdusert før eller senere, også når det er helt unødvendig. Det som skjer da, har jeg sett, er at de flinke bare trekker på skuldrene og fortsetter som før. Så trer ledelsen inn og håndhever metodene med sterk overbevisning. Det er da de ekte programmererne forlater skipet. Alle slike metoder tiltrekker seg dårlige ledere, sånne som tror de er smarte og vet mye, siden de er ledere. Det er en nedadgående spiral, ofte forbundet med at bedriften blir for stor. Det blir som det alltid blir: Gjennomsnittlig på alle måter. Ved å bruke gjennomsnittlige metoder kan dårlige - beklager - gjennomsnittlige programmerere utvikle gjennomsnittlige produkter. Og motta gjennomsnittlig lønn.

Jeg trodde, når smarttelefoner begynte å dukke opp, at det ville føre til en ny tidsalder med datahobbyisme som på 80-tallet, med at folk hacket, delte kode og ideer og hadde det gøy. Med uendelig informasjon tilgjengelig, så ville folk sikkert prøve å finne opp sine egne ting? Ikke så fort! Apple låste telefonene sine. Hvis du vil skrive kode for din nye iPhone, er det en komplisert registreringsprosess; mer penger koster det også. Og alt er ikke bare fryd og gammen hos Google heller: Android sine verktøy og prosjekter er komplekse nok til å holde unge programmerere langt unna. Kort fortalt, ingen brydde seg, så nå brukes smarttelefoner stort sett til lett underholdning og for å se hva vennene gjør.

[1] https://medium.com/@Kieran11/the-importance-of-revisiting-your-old-code-58eeb93a0dd7

Personlig eier jeg ingen smarttelefon, og har ingen planer om å skaffe en heller. Jeg liker mekaniske klokker. Ikke fordi de er nøyaktige, noe de slett ikke er, men fordi de minner meg om kunsten å programmere. Akkurat som i prosessering av store datavolumer må hver lille del gjøre jobben sin perfekt og til rett tid. Det er ikke rom for noe slark i et eneste tannhjul. Det er den typen programmering som beskrives i denne boken, og som sauene skyr vekk fra: Ekte programmering.